

hors série

Led MICRO

PROGRAMMATION COURS 2^{ème} CYCLE

COURS

N°33

Suite
2^e cycle

N°13

COURS DE
PASCAL
la notion
de type

COURS DE
PROGRAM-
MATION
APPROFONDIE :
mise au point
d'un programme

COURS
D'INITIATION AU
PROGICIEL
MULTIPLAN



ISSN 0757-6189

VOYAGE AU CŒUR DES MICRO-ORDINATEURS

dans la
COLLECTION
"ETUDES"
aux
éditions
fréquences



**une véritable
schémathèque**

- 128 pages
 - 101 schémas
 - 34 tableaux
- Prix : 150 F**

Que ce soit pour concevoir des interfaces ou optimiser un programme (utilisation des périphériques, encombrement mémoire...) «un micro-informaticien performant» doit posséder une bonne connaissance de son matériel.

Ce livre s'adresse donc à tous les électroniciens qui désirent découvrir les différents

composants constituant un micro-ordinateur. Articulé autour du microprocesseur Z80, cet ouvrage contient de nombreux schémas (plan mémoire, interfaces série et parallèle, interface clavier, interface vidéo, CAN, CNA...) qui pourraient être le thème... de nouvelles extensions.

En vente chez votre libraire et aux Éditions Fréquences

BON DE COMMANDE

Je désire recevoir l'ouvrage *L'électronique des micro-ordinateurs* au prix de 150 F (150 F + 10 F de port)

Nom

Adresse

A adresser aux ÉDITIONS FRÉQUENCES 1 boulevard Ney, 75018

Paris

Réglement à joindre

Par chèque bancaire ☐ par chèque postal ☐ par mandat ☐

Philippe Faugeres, Docteur-ingénieur en électronique a acquis son expérience dans de grandes entreprises françaises où pendant cinq ans, il a travaillé sur des systèmes d'automatismes à base de microprocesseurs. Philippe Faugeres est responsable de la rubrique «Raconte-moi le micro-informatique» dans la revue LED.

hors série Led MICRO

PROGRAMMATION COURS 2^e CYCLE

Société éditrice
Éditions Frequentes
Sage éditeur
1, bd Ney - 93012 Paris
Tél. : 01 48 01 01 81
SA au capital de 1 000 000 F
Président-Directeur Général
Edouard Pivator

LED MICRO
Cours 2^e cycle
Méthode : 10 F
Commission paritaire : 0045
Directeur de la publication
Edouard Pivator
Tous droits de reproduction réservés
tous et pour tous pays
LED MICRO est
une marque déposée (024 013 1015)

Service Publication-Publicité
Abonnements
1, bd Ney - 93012 Paris
Tél. : 01 48 01 01 81
Logis goudrons

Comité de rédaction :
Dominique Chastagner
Jean-François Coblentz
Charles-Henry Delaleu
Patrick Gueneau

Secrétaire de Rédaction
Christel Cauchon

Publication : le mardi
Tél. : 01 48 01 81
Secrétariat et abonnements
Avis à Paris

Abonnements
10 numéros par an
France : 150 F
Etranger : 240 F

Rédaction
Concept-En-Progrès
Ed-Systeme
Impression
Beyeler-Lesneul - Nancy



OCTOBRE 86

COURS DE PASCAL de la page 5 à la page 16

- La notion de type p. 6
 - Introduction
 - Comment cela marche-t-il ?
 - Les types standard
 - Les types non standard
 - Plus sur les types non standard
 - Les autres types
 - Conclusion
- Comge de l'exercice du mois
demar p. 12
- Exemple d'un environnement
Pascal p. 16

Dominique Chastagner
Jean-François Coblentz
Patrick Gueneau

COURS DE PROGRAMMATION APPROFONDIE

de la page 18 à la page 24

- La finion p. 19
 - Mettre à jour deux versions
 - Nettoyage du programme
 - Suppression des zones de mise au point
 - Suppression des commentaires
 - Renumerotation
 - Ajout des modules déjà au point
 - Test global du programme
- Conclusion p. 23
- Annexe p. 23

Dominique Chastagner
Jean-François Coblentz
Patrick Gueneau

DIALOGUE AVEC NOS LECTEURS de la page 26 à la page 32

C'EST ARRIVÉ DEMAIN pages 36 et 37

COURS D'INITIATION AU PRODIGE MULTIPLAN

de la page 38 à la page 49

- La commande VERS p. 39
- La commande Guide Opérateur p. 39
- La commande TAB p. 39
- La commande Quitte p. 39
- La commande Alpha p. 40
- La commande largeur
de colonne p. 40
- La commande calcul p. 41
- La commande format p. 42
- Les sauvegardes p. 44
- La commande Insere p. 44
- Le caractère « » p. 45
- La commande blanc p. 45
- La commande recopie p. 46
- Expression - Calcul p. 47

Charles-Henry Delaleu

NOTRE COUVERTURE : C'est la révolution dans le monde des PC. Amstrad qui était un des clous du dernier Scob vient de sortir un compatible, le PC-1512 à moins de 5 000 F HT en version de base. Certains ont dit miracle, d'autres (ses concurrents) parlent de... A suivre.

Microprocesseurs un cours essentiellement pratique !



Philippe Duquesne, ingénieur électronique (I.S.E.N.) est chargé du cours de microprocesseurs au C.N.A.M. de Paris. Depuis plus de dix ans, il a pris goût à l'enseignement et il est l'auteur d'un ouvrage didactique sur l'électronique digitale et notamment d'un cours pratique de microprocesseurs. Fervent pratiquant du « dialogue » école/industrie, après avoir exercé les fonctions de chef de département électronique chez Burroughs, second constructeur mondial en informatique, il est actuellement chef du service Études Électroniques au sein de la direction technique chez Messier Hispano Bugatti (groupe SNECMA) avec pour principal objectif l'introduction des microprocesseurs dans les avions d'atterrissage.

Diffusion auprès des libraires assurée exclusivement par les Éditions Eyrolles

Pour ceux qui veulent aborder la micro-informatique en désirant en connaître les éléments essentiels ; ceux pour qui la « puce » ne doit pas rester un mythe.



Electronique digitale ?

Notre temps aura témoigné d'une nouvelle technique, une autre façon de communiquer avec l'électronique digitale.

Philippe Duquesne, professeur chargé de cours au CNAM, a su dans cet ouvrage en expliquer clairement les fondements.



Bon de commande

à adresser aux ÉDITIONS FREQUENCES 1, bd Ney 75018 PARIS

Je désire recevoir le(s) ouvrage(s) suivant(s) :

☐ INITIATION A L'ELECTRONIQUE DIGITALE au prix de 105 F (95 F + 10 F de port)

☐ INITIATION AUX MICROPROCESSEURS au prix de 105 F (95 F + 10 F de port)

Ci-joint mon règlement par : ☐ CCP ☐ Chèque bancaire ☐ Mandat

Nom Prénom

Adresse Ville

Code postal Ville

COURS DE PASCAL

Dominique Chastagnier
Jean-François Coblenz
Patrick Gueneau

Nous abordons maintenant un des domaines qui font des langages évolués, dont le Pascal fait partie, un miracle permanent. Il semble en fait que ce soit plus exactement une sorte de jonglerie miraculeuse, toujours sans filet, mais sans jamais chuter.

Jusqu'à présent, les seuls types de données bien connues que nous ayons utilisés sont : REAL, INTEGER, BOOLEAN, CHAR. Nous allons voir qu'il est possible de créer toutes sortes de types, entièrement adaptés aux besoins du moment. Nous avons déjà effleuré ces possibilités, mais nous tenterons d'aller bien au fond des choses dans ce cours.

COURS N° 4

PLAN DU COURS

1. La notion de type
 - 1.1 Introduction
 - 1.2 Comment cela marche-t-il ?
 - 1.3 Les types standard
 - 1.4 Les types non standard
 - 1.5 Plus sur les types non standard
 - 1.5.1 Les sous-types
 - 1.5.2 Les types intervalles et la composition des données
 - 1.6 Les autres types
 - 1.7 Conclusion
2. Corrigé de l'exercice du mois dernier

En fait, il est possible d'adapter tout programme, toute syntaxe Pascal, à des besoins spécifiques, lors de la programmation. Ainsi, on gagne en général de la place, de la vitesse d'exécution, de la vitesse de modification du programme grâce à un écrasement de la lisibilité. Ces points, nous l'avons vu depuis la création de Lcd-Micro sont ceux qui font que l'on programme avec plaisir et efficacité. Par exemple, nous verrons comment créer des types particuliers de variables, comme un type `Jours_de_semaine`, ce qui est bien commode.

1. LA NOTION DE TYPE

1.1. Introduction

La notion de type est un des principes de bases du Pascal. Il est indispensable de bien comprendre comment ils sont gérés pour utiliser pleinement les possibilités du langage. Le Pascal est très rigoureux pour la gestion de ces types, ce qui permet un potentiel de fonctions très large, avec une sécurité à peu près totale. Nous allons par la suite revenir sur les types standards, les types prédéfinis, et les types créés par l'utilisateur.

1.2. Comment cela marche-t-il ?

Tout d'abord, il est important de savoir que chaque «type» est stocké différemment, afin d'optimiser la place. En effet, par exemple, il serait dommage de stocker un entier en gardant de la place pour le point décimal, et les chiffres situés à droite de ce point. Cette optimisation relative a déjà été rencontrée en Basic, et elle est bien sûr présente ici. Mais elle est aussi poussée à un plus haut niveau de sophistication. Pour cela, la définition de types est fondamentale, car elle annonce au système comment il faudra stocker l'information. Chaque type a sa structure de stockage, ce qui est bien commode sur le plan de la place économisée en mémoire et sur disque.

1.3. Les types standard

Les types standard du Pascal sont :

- **INTEGER** : les entiers dits «relatifs», positifs et négatifs. Il s'agit d'un type énuméré, c'est-à-dire qu'ils forment une suite énumérée, et certaines commandes sont alors disponibles, comme **PRED** et **SUCC**, qui donnent respectivement le prédécesseur et le successeur d'un élément. Ces commandes, peu utiles ici, seront indispensables ailleurs.
- **REAL** : les nombres, avec décimales, couvrant la totalité des types de nombres. Le type n'est plus énuméré, et peu de commandes sont spécifiques à ce type.
- **CHAR** : les caractères de l'alphabet, plus les caractères dits spéciaux, qui sont les `^`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`, `^[`, `^]`, `^_`, `^@`, `^A`, `^B`, `^C`, `^D`, `^E`, `^F`, `^G`, `^H`, `^I`, `^J`, `^K`, `^L`, `^M`, `^N`, `^O`, `^P`, `^Q`, `^R`, `^S`, `^T`, `^U`, `^V`, `^W`, `^X`, `^Y`, `^Z`,

Regardez si ce programme marche, et si non, ou il plante. A vous de le modifier pour qu'il marche et bien comprendre le fonctionnement de la structure d'énumérés.

Integer

Réel

Booleen

Caractère

1.4. Les types non standard

Nous entrons dans le domaine où l'imagination est au pouvoir. Il est possible de définir n'importe quel type de données, que ce soit à partir de types existants, ou de types complètement nouveaux. Un exemple simple et fatidique, sauf pour les professions dentaires peut-être.

```
type
  dents = (incisive, canine, molaire);
```

Un autre, plus bête, à partir d'un type existant, INTEGER :

```
type
  zero_deux = 0..2;
```

Fait des entiers entre 0 et 2, donc 0, 1 et 2.

Un type pour les comptables, ou ceux qui programment un calendrier :

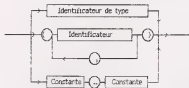
```
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi,
  dimanche);
```

ou encore :

```
type
  jour_travail = lundi .. vendredi;
```

Du lundi à vendredi, dans le type prédéfini, jours.

Définition d'un type simple



Ce petit diagramme montre comment construire un type simple, depuis les structures dont vous avez besoin.

Il est donc possible de comprendre que c'est ici que se traite une grande partie de la lisibilité des programmes Pascal, et leur rapidité, car ceci permet au système de savoir très vite ce qu'il aura à utiliser, et sous quelle forme.

De plus, vous pourrez avoir alors un programme comme suit :

```

program Jours;
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche);
var
  j:jours;
begin
  for j:= lundi to vendredi do writeln(j);
end.
  
```

Jok, non ? Mais surtout très clair, pour le programmeur et d'autres personnes prenant le listing pour la première fois. Ce listing est lisible, mais aussi documenté directement, puisqu'il remplace des commentaires du type :

«travail pour chaque jour de la semaine»

Nous vous le disons, ce langage est splendide !!!

La raison pour laquelle le morceau de programme fonctionne est que nous venons de définir un type **énuméré**, du nom de JOURS. Faites donc tourner le programme suivant :

```

program Jours;
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche);
var
  j:jours;
begin
  for j:= mardi to samedi do writeln(j, ' ', pred(j), ' ', succ(j));
end.
  
```

Impressionnant

Pour faire un calendrier, il serait possible de définir en plus les types mois :

```

mois = (janvier, fevrier, ..., decembre);
  
```


et numero du jour du mois :

```
num_jour = 1..31;
```

Que c'est beau, je suis tellement ému, j'en pleure!

Mais attention, il est sûr que le système n'acceptera pas n'importe quoi. Il est évident qu'il ne peut accepter des informations contradictoires, et pour cela doit traquer sans pitié tout ce qui pourrait l'être ou simplement le devenir.

Par exemple, le programme suivant plante dès la compilation, c'est promis :

```
program Jours;
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche);
  mois = (janvier, fevrier, mars, avril, mai, juin, juillet, aout,
          septembre, octobre, novembre, decembre);
  num_jour = 1..31;
var
  j:jours;
  m : mois;
  n : num_jour;
begin
  for m:= mardi to samedi do writeln(j, pred(j), succ(j));
end.
```

m est défini comme variable de type *mois*, et ne peut prendre ses valeurs dans le type *jours*. Donc, pas de confusion, la rigueur est toujours présente, et même plus que jamais. C'est pourquoi un certain nombre de points doivent être clairement expliqués. Par exemple, une donnée définie dans un type ne peut pas l'être ailleurs, autrement. Cela semble stupide, mais regardons l'exemple que nous avons fabriqué exprès pour vous :

```
program Jours;
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche);
  travail = (lundi, mardi, mercredi, jeudi, vendredi);
  mois = (janvier, fevrier, ..., decembre);
  num_jour = 1..31;
var
  j:travail;
  m : mois;
  n : num_jour;
begin
  for j:= mardi to jeudi do writeln(j, pred(j), succ(j));
end.
```

Ce programme ne marchera pas, car les jours ouvrables sont définis sous deux types différents, *jours* et *travail*. Vous pouvez voir ici que parfois redéclarer une donnée dans deux types peut arriver. Or, c'est au programmeur de vous avertir lorsque vous donnez par mégarde deux fois une définition, qu'elle vous semble logique ou non. Pour permettre malgré tout des possibilités comme celles requises, nous verrons comment définir des sous-types ou des intervalles. Donc attention aux redéclarations qui peuvent vous sembler commodes, ou vous échapper.

C'est interdit

Alors, il faut être attentif, sinon le système le sera pour vous à la compilation, souvent sans grande politesse, mais avec efficacité.

1.5. Plus sur les types non standard.

1.5.1. Les sous-types.

Nous venons de dire qu'il est interdit de redéclarer des données dans des types. Nous vous avons également annoncé que nous venions comment résoudre ce qui peut passer pour un problème.

Il est en effet possible de définir un type comme un sous-type d'un type déclaré ou standard. Dans le programme JOURS, nous avions par exemple la déclaration suivante :

```
program Jours;
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche);
  mois = (janvier, fevrier, ..., decembre);
  num_jour = 1..31;
```

Vous le voyez ici, num_jour est un sous-type INTEGER. De la même façon, nous pouvons résoudre notre problème de jours ouvrables de la manière suivante :

```
travail = lundi..vendredi;
```

Ceci permet de définir Travail comme un sous-type de jours, défini comme un intervalle d'un type énuméré. Il n'y a plus de conflit de type, puisque TRAVAIL est donc ainsi comme une partie de JOURS, un sous-ensemble, qui est ici un intervalle de ce type.

On a alors :

```
program Jours;
type
  jours = (lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche);
  mois = (janvier, fevrier, ..., decembre);
  travail = lundi..vendredi;
  num_jour = 1..31;
```

Ceci permet de remettre en place des possibilités qui semblaient interdites en raison des définitions de types.

1.5.2. Les types intervalles, et la compaction des données.

Il est parfois très intéressant d'utiliser les sous-types, car cela permet aussi dans certains cas de gagner de la place en stockage. Ainsi, déclarer des variables de type 0..1, donc prenant 0 ou 1 comme valeur uniquement, permet de dire au programmeur que dans certaines circonstances que nous décrirons plus loin, ces données pourront être stockées sur un unique BIT, d'où un impressionnant élagage de la taille, car le stockage de données numériques entières est généralement fait sur 16 BITS, deux octets. Le gain est appréciable.

1.6. Les autres types.

Supposons que vous écriviez le programme

```

program essai(input,output);
  var toto : integer;
begin
  writeln(' donnez un nombre entier entre 1 et 10 ');
  readln(toto);
  ...

```

Si par la suite votre programme attend une donnée vraiment entre 1 et 10 pour des traitements particuliers pour chaque valeur, et que par erreur vous voyez entre 11, que faire ? Une bonne protection est de vérifier le résultat, et de recommencer si le nombre entre est différent de ce qui est attendu. Pour cela, il y aurait bien sûr GOTO, pour recommencer la boucle jusqu'à satisfaction du test. Mais il existe également d'autres méthodes de branchement, et une méthode de test plus élégante, utilisant les notions ensemblistes. Il est possible de tester l'appartenance d'une variable à un ensemble, et de jouer sur cette appartenance dans un test.

Il est en effet possible de programmer en indiquant que l'on attend une donnée appartenant à un tel ensemble. Pour cela, la commande est simple. Pour reprendre le cas précédent, le programme peut être modifié de la manière suivante, donnant une écriture très naturelle de la condition :

```

program essai(input,output);
  var toto:integer;
begin
  repeat
    writeln(' donnez un nombre entier entre 1 et 10 ');
    readln(toto);
  until toto in [1,2,3,4,5,6,7,8,9,10];
  ...
end.

```

Ce programme boucle autant de fois qu'il le faudra pour obtenir le nombre désiré. Mais il peut être rendu plus agréable à taper et à lire. En effet, le moins que l'on puisse dire est que si vous voulez un nombre entre 1 et 1 000 000, la frappe sans longue. Alors regardez la suite :

```

program essai(input,output);
type
  nombre_attendu = 1..1.000.000;
var
  toto:integer;
begin
  repeat
    writeln(' donnez un nombre entier entre 1 et 1.000.000 ');
    readln(toto);
  until toto in nombre_attendu;
  ...

```

Il semble difficile de faire plus simple, compact et lisible. En fait, structurer un programme devient un plaisir avec de tels outils.

1.7. Conclusion.

Plus encore que pour tout le reste, la seule façon de bien maîtriser la puissance des structures que nous avons vues ici est de programmer en s'en servant. Il est toujours possible de ne pas utiliser les types, mais leur utilité, nous l'espérons, ne vous aura pas échappé. Nous continuerons le mois prochain sur ce point, afin de montrer comment utiliser au mieux les déclinaisons de type. De plus, nous vous proposerons alors des exercices mettant en valeur la puissance du procédé.

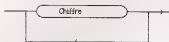
2. CORRIGE DE L'EXERCICE DU MOIS DERNIER.

Le mois dernier, nous vous avons proposé de réaliser les diagrammes syntaxiques des principales structures déjà rencontrées.

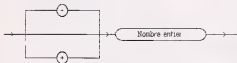
Les voici, dans un ordre croissant de difficulté ou approximativement, car ce n'est de toute manière pas trop complexe.

Constante :

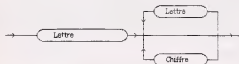
Nombre entier



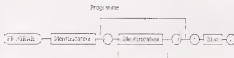
Constante entière



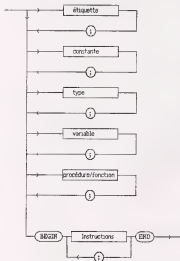
Identificateur



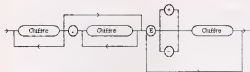
Note : Un identificateur est un nom donné à une structure, variable, programme, sous-programme, constante, type...



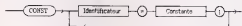
Un bloc est ce qui suit l'en-tête du programme. Il est donc de la forme suivante :



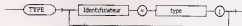
Description des constantes numériques :



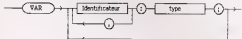
Déclaration des constantes



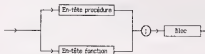
Déclaration des types



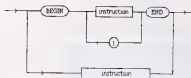
Déclaration des variables



Déclaration de procédures ou de fonction



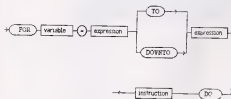
Instruction composee



Test



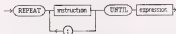
Première boucle



Autre boucle



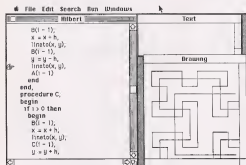
Troisième boucle



Il y a naturellement, d'autres diagrammes à envisager, mais ceux-ci représentent la plus grande partie de ce que nous avons déjà vu. Par la suite, lors de l'étude de certaines structures, et pour mieux les faire comprendre, nous reprendrons ces diagrammes. Parmi les diagrammes simples que nous avons délaissés, trouvez-vous ceux des instructions ou blocs d'instructions, ou de déclarations, les plus utilisés ? À vous de les représenter. Si d'importants diagrammes nous ont échappés, nous essaierons de les retrouver durant ce mois, avec votre aide.

Le mois prochain, nous traiterons des types, pour conclure sur cette notion, puis nous commencerons l'étude des structures de contrôles, IF, REPEAT et WHILE. Celles-ci sont généralement bien connues de ceux d'entre vous qui ont suivi le cours de BASIC. Puis nous présenterons une structure fort intéressante, et nouvelle, le CASE. Mais pour vous dire plus, il vous faudra languir un mois. Nous vous «flançons» le cœur ? Nous serons employables, qu'on se le dise.

EXEMPLE D'UN ENVIRONNEMENT PASCAL



PASCAL interactif sur l'ordinateur MACINTOSH

TOUT SUR LES PÉRIPHÉRIQUES



- ☐ 85 schémas
- ☐ 20 tableaux
- ☐ 136 pages
- Prix : 150 F

Les périphériques font partie intégrante d'un système informatique. En parallèle de l'unité centrale, qui gère et synchronise l'ensemble, ils sont responsables de différentes fonctions comme :

- la mémoire de masse : unités de disques souples et de disques durs, lecteur de cassette ;
- le dialogue avec l'utilisateur : clavier, écran vidéo, imprimante ;
- les télécommunications : modem

Tous ces périphériques sont décrits dans cet ouvrage avec, pour chacun d'eux, une partie technique (principe de fonctionnement, caractéristiques techniques) et une partie interface (souplesse d'entrées-sorties, connecteurs de liaison).

Dans chaque grande catégorie (memories, imprimantes) une analyse comparative des différents produits existants est effectuée.

Philippe Feuguères, docteur ingénieur en électronique, est responsable matériel dans une entreprise d'informatique traitant des réseaux de PC. Au préalable, il a acquis une expérience en travaillant sur des sujets comme les automatismes et les télécommunications dans deux grandes sociétés françaises (Bull, CGE). Philippe Feuguères est l'auteur d'un premier ouvrage « L'Électronique des micro-ordinateurs » paru aux Éditions Freyriennes.

BON DE COMMANDE

Diffusion auprès des libraires auteurs exclusivement par les Éditions Cyrolis

Je désire recevoir l'ouvrage «Périphériques interfaces et technologie» au prix de 162 F (150 F + 12 F de port)

Nom

Adresse

Réglement à joindre :

Par chèque bancaire ☐ par chèque postal ☐ par mandat ☐

COURS DE PROGRAMMATION APPROFONDIE

Dominique Chestagnier
Jean-François Coblenz
Patrick Guaneu

Nous abordons aujourd'hui le dernier volet de la série «mise au point d'un programme». Après plus d'une année d'articles consacrés à des sujets comme l'optimisation des performances en BASIC, la structuration des données et maintenant la mise au point, vous disposez de bonnes bases pour vous lancer dans le développement de votre propre application... A vos claviers !

COURS N° 13

PLAN DU COURS

La mise au point des programmes - Finition

- 1.1 Mettre à jour deux versions
- 1.2 Nettoyage du programme
- 1.3 Suppression des zones de mise au point
- 1.4 Suppression de commentaires
- 1.5 Renumerotation
- 1.6 Ajout des modules déjà au point
- 1.7 Test global du programme
 - Cohérence entre les modules
 - Dimensionnement en grandeur réelle des données

Retour sur le mois précédent

Avant de passer à la troisième et dernière étape de cette série, nous voulons apporter quelques éclaircissements sur un chapitre du numéro précédent. En effet, les commentaires du chapitre 1.1.1.1 sur la figure n'ont tout simplement pas lieu d'être. Nous espérons que cette erreur ne vous aura pas trop causé de maux de tête. De toutes façons, nous reviendrons prochainement sur ces méthodes par des exemples pratiques (mois de décembre).

1. LA FINITION

Les recommandations de cette troisième partie vont sûrement apparaître superflues pour certains d'entre vous. Cependant, et bien que cette série d'articles soit quelquefois allée bien loin dans les détails, nous voulons apporter au programmeur débutant un certain nombre de repères et de réflexes pour qu'il programme mieux, plus vite et avec plus de méthode et ce, dès ses premiers pas (le programme bien sûr). Inversement, il nous est parfois difficile de bien appréhender les difficultés que peuvent rencontrer nos lecteurs dans leur progression, aussi cette série a-t-elle dû sembler trop ardue à d'autres. Encore une fois, écrivez-nous pour nous préciser les domaines où vous éprouvez le plus de difficultés, nous vous répondrons soit dans la rubrique programmation approfondie, soit dans la rubrique «dialogue avec nos lecteurs». Cette dernière partie vous sera d'autant plus profitable que vous l'appliquerez au bon moment, après la mise au point. C'est pourquoi nous l'avons séparée de cette dernière. En réalité, puisqu'il y a modification du programme, il y a risque d'erreur donc de corrections. Vous trouverez donc des conseils pour véritablement achever votre travail.

1.1. Mettre à jour deux versions

Il est le plus souvent primordial de disposer de deux versions d'un même programme : une version complète (commentaires et zones de mise au point), et une version nettoyée et compactée utilisée pour les tests finaux. Il n'y a aucune difficulté technique réelle à condenser des lignes BASIC mais par contre il faudra régulièrement reprendre ce travail de manière à conserver une évolution parallèle des deux versions. On peut se demander en effet à quoi servent des commentaires qui ne correspondraient plus à la version «compactée».

1.2. Nettoyage du programme

Vous voilà donc avec une application qui «fonctionne» convenablement. Bien entendu, grâce aux ajouts successifs des modules de mise au point, des correcteurs diversifs, voire des débrouillages savants, votre listing ne ressemble plus à rien (qui a dit que l'informatique était remplacer le papier et le crayon !), le nombre de versions de votre programme a dépassé le dixième (et peut-être plus), ou pire, ce n'est qu'après chargement, à la lecture de zones révélant des listings que vous savez à quelle version vous avez affaire. Quand on est informaticien à ses heures perdues, il est difficile de se souvenir de la petite retouche effectuée au dernier moment juste avant d'aller se coucher (surtout que le lendemain on n'est plus trop sûr de l'avoir enregistrée).

Il faut donc agir méthodiquement. Pour vous aider, nous vous proposons une liste des tâches à effectuer pour bien conclure le développement de votre application :

- isolez la bonne version du programme,
- créez un support distinct (disquette ou cassette) pour sauvegarder cette version,
- travaillez toujours en l'utilisant comme référence, si possible gardez une trace papier (à propos, si vous envisagez un investissement pour votre machine, deux extensions s'imposent : une ou mieux deux unités de disquette et ensuite une imprimante, elle vous rendra de grands services),
- effectuez régulièrement des sauvegardes d'une part des modifications mais aussi référez les phases précédentes afin de conserver à tous moments un environnement proche des versions en cours.

1.3. Suppression des zones de mise au point

Il y a seulement deux risques majeurs lors du nettoyage du votre programme. Le premier est en fait équivalent à la suppression des commentaires (cf. plus loin) et ne

présente guère de danger. Par contre, il arrive que les modules de mise au point cachent des erreurs de programmation, ou faussent un résultat sans qu'on s'en rende compte. Ce n'est qu'au test global du programme que l'erreur apparaît puisque ces modules ont disparu. Ce peut être, par exemple, une initialisation effectuée dans un de ces modules et pas dans le programme même. On conçoit aisément que la précaution avec laquelle on aura ajouté ces lignes de «débogage» (c'est le nom français pour «debugging») influencera les risques d'erreurs à ce niveau. Comme vous le voyez, tout se tient.

1.4. Suppression de commentaires

La suppression des REMs ne présente guère de risques. Cependant, faites attention aux appels GOTO et GOSUB référencant des lignes commentées, il est effectivement courant de commencer une zone de programme ou sous-programme par des commentaires (en tout cas, nous vous y encourageons). L'exemple suivant correspond aux erreurs les plus fréquentes.

COMPACTAGE DE PROGRAMME

```
10 REM un programme simple avant compactage
15 REM
20 INPUT "NOMBRE":NB
25 IF NB<0 THEN GOTO 20
30 FOR I=1 TO NB
40 PRINT "Le Code ASCII ",NB,
45 PRINT "donne le caractère ",CHR$(I)
50 NEXT I
60 STOP
70 END
```

AVANT COMPACTAGE

Suppression des REMs

Regroupement sur une ligne
de plusieurs instructions

```
10 INPUT "NOMBRE":NB:IF NB<0 THEN 10
20 FOR I=1 TO NB:PRINT "Le Code ASCII ",NB,"donne le caractère ",CHR$(I)
NEXT:STOP:END
```

APRES COMPACTAGE

ERREURS DE COMPACTAGE

10 REM 1er exemple de compactage excessif !!
20 INPUT "NOMBRE":NB:IF (NB<0) THEN NB=0:FOR I=1
TO NB

10 REM 2ème exemple de compactage excessif !!
20 INPUT "NOMBRE":NB:IF (NB<0) THEN GOTO 20:FOR
I=1 TO NB

REMARQUES:

A force de regrouper les lignes, on en oublie qu'on ne peut avoir aucune instruction après un GOTO, car elles ne seront jamais exécutées. Une deuxième erreur fréquente est de regrouper des instructions derrière un test. Attention tout ce qui suit le THEN n'est réalisé que si le test est vrai, et non pas seulement la première instruction.

1.5. Renumerotation

Attention, cette manœuvre est délicate, et d'autant plus si les possibilités d'édition de votre interpréteur BASIC sont limitées (n'est-ce pas, heureux possesseurs d'Apple II ?). En effet, que faire par exemple lorsque vous voulez insérer une ligne entre 100 et 101, sinon décaler la ligne 101, etc., jusqu'à ce qu'il y ait de la place. En conclusion, une renumérotation de 10 en 10 vous assurera des éventuelles dernières retouches.

1.6. Ajout des modules déjà au point

Le test complet du programme faisant partie de la finition du projet (c'est en tout cas notre avis), on ajoute donc les modules complémentaires qui manquent à l'application, que ces modules aient déjà été vérifiés auparavant dans la phase de mise au point, ou issus d'autres applications. Fréquemment, ces modules sont des gestionnaires d'écrans ou des utilitaires (fichiers, impression) et ne sont pas nécessaires au bon fonctionnement du programme.

1.7. Test global du programme

Pourquoi considérer le test global du programme comme une finition du développement plutôt qu'une phase de mise au point ? Il y a trois raisons essentielles :

- Si vous respectez les conseils que nous vous avons donnés le mois dernier, vous n'aurez en fin de mise au point pas encore pu tester le programme en entier, à cause des modules supprimés des tests tentés aux seuls cas simples, etc., mais vous aurez corrigé la plupart des erreurs internes à chacun des modules indépendants sans que les modules principaux (par exemple, si vous réalisez un programme de gestion de fichiers, ces modules seront ceux traitant une option du menu principal, création de fichiers, listes, ajouts, modifications, etc.) aient, si votre analyse est correcte, rien d'irremédiable ne pourra tout remettre en cause.
- Il faut éviter le plus longtemps possible de dépendre des limitations de l'environnement autres que celles de l'analyse et du langage utilisé, plus simplement c'est à l'analyse de préciser le plus clairement possible dès le départ quelles seront vos contraintes externes. Ces limitations sont donc en théorie connues avant même la phase de programmation (ainsi, en reprenant l'exemple précédent, vous savez qu'en considérant les capacités de votre lecteur de disquette vous ne pourrez traiter plus d'un certain nombre d'enregistrements par fichier, etc.).
- Enfin, si la phase précédente (« mise au point ») a été bien menée, une erreur importante révélera presque à tous les coups une lacune sérieuse dans la conception du programme, entraînant un travail beaucoup plus important que la plupart des corrections apportées dans cette mise au point, en fait, ce travail est d'autant plus pénible que la détection de l'erreur est très délicate à ce niveau et qu'il est parfois nécessaire de tout reprendre à zéro.

Cette ultime étape décide ainsi du sort que vous allez réserver à votre programme et il n'y a pas deux alternatives, ça passe ou ça casse. Admettez qu'il n'est pas très agréable de se retrouver après des dizaines d'heures de travail face à un programme qu'il faut jeter à la poubelle ! Voici maintenant des détails des vérifications à effectuer :

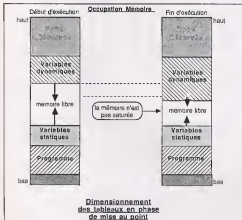
1.7.1. Cohérence entre les modules

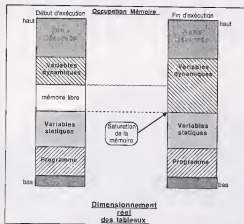
Rassurez-vous, cela signifie simplement qu'il faut essayer chacune des fonctions de votre application pour vérifier si leur indépendance est réelle. Le plus simple est alors de procéder à une session complète (ou mieux de soumettre le programme aux essais imprévisibles d'autres personnes de votre entourage, les candidats n'ont pas leur pareil pour mettre en défaut en quelques secondes un programme sur lequel vous souffrez depuis fort longtemps), de comparer les résultats avec les tests préliminaires que vous aurez conservés de la phase de mise au point. Pour que vous puissiez qualifier votre application de fiable, il faudra user et abuser de tests multiples, reprendre des cas particuliers.

Ainsi, toujours à l'aide de l'exemple d'une gestion de fichier, exécuter une session complète implique l'accès à toutes les rubriques du menu principal, c'est-à-dire création du fichier, insertion d'enregistrements, sélection d'une partie de ces enregistrements pour modification et pour impression, etc.

1.7.2. Dimensionnement en grandeur réelle des données

Vous sentez maintenant que vous approchez du but, toutes les fonctionnalités sont opérationnelles, c'est le moment de reprendre les déclarations de taille de tableaux, les allocations sur disques pour vérifier les limites fixées ou peut-être essayer de faire mieux. En fait, c'est l'interaction entre le programme et l'environnement qu'il vous reste à analyser. Dans la pratique, les risques de surprises sont inversement proportionnels à la finesse et à la qualité de l'analyse. Malgré tout, et surtout parce qu'on ne peut connaître la taille du programme qu'une fois celui-ci terminé, il est difficile d'estimer avec précision et assurance l'encombrement en mémoire de l'ensemble programme + données, le compactage du programme est une bonne solution pour être le meilleur parti de ce que le BASIC vous autorise en mémoire libre.





CONCLUSION

Le plus délicat lorsqu'on termine une application est de conserver l'âme du programme tout en compilant tout défaut fatal à son exécution, mais aussi de maintenir ce parallélisme entre la version complète, commentée et celle optimisée, compactée, qui est exécutée. Cette contrainte est d'ailleurs spécifique aux interpréteurs, nous verrons plus tard qu'en PASCAL notamment, à la compilation, les zones de commentaires sont supprimées puisque non exécutables. Alors, prenez votre mal en patience si vous travaillez en BASIC interprété.

Cette série se termine donc, il nous restera à vous montrer plus pratiquement comment procéder à l'aide d'exemples simples (un mini-traitement de texte sera qu'un jeu de réflexes). Cependant, nous étudierons en pronto le mois prochain les différences entre compilation et interprétation, tant au niveau du BASIC qu'à celui du PASCAL, afin de vous laisser le choix en toute connaissance de cause. Cette étude permettra aux futurs «pascalons» de mieux connaître l'environnement typique en PASCAL qui est d'ailleurs fort différent de celui de la plupart des BASICs, et pour ceux qui hésitent encore, ils trouveront peut-être la une bonne raison qui les fera basculer dans le camp des programmeurs PASCAL.

ANNEXE





Depuis maintenant un an, nous avons abondamment disserté dans la partie «programmation approfondie» des méthodes d'optimisation dans la structure et le

traitement des données, les temps de calculs et d'exécution, le compactage ou encore la précision de calculs. Lorsque votre application réalise ce que vous voulez qu'elle fasse, et si vous en avez le courage, compléter la mise au point par une phase ultime d'optimisation. Ce ne sera pas d'ailleurs forcément une amélioration des performances du programme, mais peut-être plus simplement l'amélioration de l'interface avec l'utilisateur, une meilleure utilisation des possibilités graphiques, la simplification de certains algorithmes et bien d'autres encore... et puis, un programme n'est jamais vraiment terminé !

COURRIER DES LECTEURS : CARRÉS MAGIQUES (suite)

Nous ouvrons à nouveau le débat des carrés magiques, suite à la lettre de M. ALLAIS qui conteste la première solution présentée pour la résolution des carrés d'ordre 4. Il semble que nous n'ayons pas assez insisté sur le raisonnement suivi par M. PEYRIN, auteur de cette solution. Il précise en effet qu'il ne s'est occupé que des solutions dont le centre est du type 1 à 3 (cf. figure jointe), tout en s'interrogeant sur l'existence de solutions où le centre serait de type 4. Nous avons publié cette analyse, parce que, bien qu'incomplète (la recherche d'une solution de type 4 étant bien plus difficile à programmer), elle apportait de nombreuses solutions en tenant compte des symétries et rotations. Donc bien entendu, non dans le raisonnement suivi ne prouve, ni même l'existence d'autres solutions. D'après les résultats obtenus par M. ALLAIS, ces solutions existent. C'est donc avec plaisir que nous les étudions et que nous les publierons (après les avoir transcrites en BASIC, et pourquoi pas en PASCAL, s'il nous le permet bien entendu ! Enfin, si quelqu'un a recensé le nombre de carrés 4x4 obtenu avec la seconde méthode proposée dans ce même numéro (numéro d'août), nous pourrions encore compléter l'exploration de ce sujet passionnant.

CARRE 4x4 (centre du carré)

 <p>Type 1</p>	 <p>Type 3</p>
 <p>Type 2</p>	 <p>Type 4 jamais 17</p>

Les Types 1 à 3 sont les plus faciles à programmer

Music Vidéo

Systèmes

*Chaque mois : tout sur la Musique
électronique, le standard MIDI,
le home studio et
la production
vidéo
personnelle*



BULLETIN D'ABONNEMENT

Je désire m'abonner à **MUSIC VIDÉO SYSTÈMES** (10 numéros). France : 160 F - Etranger : 240 F,
à partir du n°.....

Nom Prénom

Adresse

envoyez ce bon accompagné du règlement à l'ordre des **EDITIONS FRÉQUENCES**

1, bd Ney 75018 Paris - Tél. 46.07.01.97

MODE DE PAIEMENT :

C.C.P. ☐

chèque bancaire ☐

Mandat ☐

DIALOGUE AVEC NOS LECTEURS

1. CORRECTION DE L'EXERCICE SUR LA C.E.E.

Avant de vous donner la solution de cet exercice, rappelons les données du problème exposé dans notre numéro de mai 1985 :

A Sinsibourg, les cinq premières maisons d'une rue sont habitées par un député du Parlement Européen. Chacun d'entre eux a peint sa maison d'une couleur différente. Ils ont tous un animal favori, une boisson préférée, jouent d'un instrument de musique et n'ont aucun goût commun.

- L'Allemand habite la première maison à gauche.
- L'Espagnol possède un chien.
- Le Grec est pianiste.
- L'Italien boit du thé.
- L'Anglais habite la maison rouge.
- Le propriétaire de la maison verte boit du café.
- La maison verte est à droite de la blanche.
- Le trompettiste joue avec son chien.
- Le guitariste habite la maison jaune.
- On boit du lait dans la maison du milieu.
- L'Allemand habite à côté de la maison bleue.
- Le violoniste boit du vin.
- Le poisson rouge est dans la maison voisine du batteur.
- Le cheval habite chez le voisin du guitariste.

Nous vous demandons alors qui élève un lapin et boit de l'eau ?

Revenons les différents éléments :

- Il y a cinq nationalités : Allemand, Anglais, Espagnol, Grec et Italien.
- Les couleurs sont : Rouge, blanc, bleu, vert, jaune.
- Les animaux : Chien, chat, cheval, poisson rouge et lapin.
- Les boissons : Thé, café, vin, lait et eau.
- Les instruments : Piano, trompette, guitare, violon et batterie.

Considérons maintenant le tableau constitué pour chaque maison :

couleur				
député				
animal				
boisson				
musique				

Étudions le nombre de choix possibles pour chacune des affirmations :

- L'Allemand habite la maison de gauche → 1 choix
- Le lait se boit dans la maison du milieu → 1 choix
- La maison verte est à droite de la blanche → 4 choix
- La maison bleue est à côté de l'Allemand
- Le poisson rouge est voisin du bretteur → 8 choix
- Le cheval est à côté du guitariste
- Tous les autres ont 5 choix possibles.

Il nous reste à placer les assertions ayant une seule possibilité et à en observer l'incidence sur les autres :

C		bleu		
D	Allemand			
A				
B			lait	
M				

Une fois l'Allemand placé, il n'y a plus qu'une seule maison voisine : la bleue (ligne 11).
Observons les choix des 11 lignes restantes :

- 2. L'Espagnol possède un chien : 4 choix (colonnes 2, 3, 4, 5)
- 3. Le Grec est pianiste : 4 choix (2, 3, 4, 5)
- 4. L'Italien boit du thé : 3 choix (2, 4, 5)
- 5. L'Anglais habite la maison rouge : 3 choix (3, 4, 5)
- 6. On boit du café dans la maison verte : 3 choix (1, 4, 5)
- 7. La maison verte est à droite de la blanche : 2 choix (3-4, 4-5)
- 8. Le trompettiste possède un canari : 5 choix (1, 2, 3, 4, 5)
- 9. Le guitariste habite la maison jaune : 4 choix (1, 3, 4, 5)
- 12. Le violoniste boit du vin : 4 choix (1, 2, 4, 5)
- 13. Le poisson rouge est voisin du batteur : 8 choix (1-2, 2-1, 2-3, 3-2, 3-4, 4-3, 4-5, 5-4)
- 14. Le cheval est voisin du guitariste : 8 choix (les mêmes)

Choisissons l'assertion entraînant le moins de choix possibles et étudions chacun d'entre eux comme une hypothèse : soit nous arrivons à une impasse, soit à la solution. On voit immédiatement que la ligne 7 n'a que deux cas possibles. Prenons le premier et observons les conséquences : il en résulte automatiquement que la ligne 6 n'a plus qu'une seule place, tout comme les 5 et 9. Ensuite, une fois placés, la 5 entraîne la 4 et la 9 provoque la 14 : il ne reste plus qu'un seul endroit pour la 12, et le seul sans boisson est l'Allemand qui boirait de l'eau.

C	jaune	bleu	blanche	verte	rouge
D	Allemand	italien			Anglais
A		cheval			
B	eau	thé	lait	café	vin
M	guitare				violin

Mais c'est alors que le problème apparaît : une seule personne possède un canari et une trompette (ligne 8). Or, l'Espagnol a un chien (ligne 2) et le Grec est pianiste (ligne 3). Quant à notre tableau, il indique que l'Allemand est guitariste, l'Italien monte à cheval et l'Anglais préfère le violon. L'hypothèse est donc mauvaise et il faut la reconsidérer.

Reprenons donc le choix initial sur la ligne 7 et plaçons les maisons verte et blanche aux colonnes 4 et 5. Les lignes 5, 6 et 9 sont toujours réduites à un choix. Elles provoquent encore 4, 14 et 12 mais cette fois-ci il y a de la place pour le canari et le trompettiste en colonne 3 chez l'Anglais ! Nous laissons la fin à votre discrétion.

C	jaune	bleue	rouge	blanche	verte
D	Allemand	Italien	Anglais		
A		cheval			
B	eau	the	lait	vin	café
M	guitare			volant	

2. JEU DE CARTES

Nous allons vous présenter en programme, un jeu de cartes auquel nous jouons lorsque nous sommes en vacances. Ce jeu turc, le «Scholar», n'est pas sans point commun avec le «Scopone Scientific» Italien (que l'on voit dans le film «L'argent de la veille»).

1^{re} Le but de ce programme est d'une part de simuler le jeu de la carte, ce que vous arriverez certainement à faire sans problème, d'autre part à programmer votre ordinateur comme partenaire. Rendez cette possibilité indépendante du jeu de la carte, c'est-à-dire que l'ordinateur ne soit pas le seul à pouvoir tirer les cartes, mais qu'on puisse aussi lui demander de jouer sur une donne que nous avons nous-mêmes choisie. Comme cela, il nous suffira de faire jouer des programmes les uns contre les autres pour connaître le meilleur.

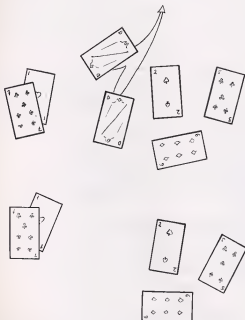
2^{de} Les règles : ce jeu se pratique avec un jeu de 52 cartes auquel on enlève les 8, 9 et 10 dans chaque couleur, cela fait donc 40 cartes. La valeur des cartes est celle indiquée par leur chiffre, l'as vaut 1, le deux vaut 2, ... le sept vaut 7, la dame vaut 8, le valet 9 et le roi 10.

Au début de la partie, on étale les 4 premières cartes du paquet puis on distribue trois cartes à chacun des joueurs qui peuvent être de 2 à 4. A quatre, les joueurs sont en équipe et se placent en vis-à-vis.

Le premier joueur regarde donc ses trois cartes et les compare avec celles étalées



Il est tenu d'étaler une carte. Toutefois, si sur la table se trouvent une ou plusieurs cartes dont la somme des points est égale à celle qui vient d'être posée, la dernière carte et celles qui constituent la somme sont restituées au joueur qui les pose devant lui.



Le joueur n'a plus en main que les cartes 7 et 1 et sur la table restent étalées le 6, 5 et le 2. Vous avez remarqué que les cartes 6 et 2 forment une somme de 8 égale à la dame. Cependant, le joueur ne doit ramasser que la somme nécessitant le minimum de cartes, dans ce cas, une (la dame) contre deux (le 6 et le 2). C'est maintenant à son adversaire de jouer.



Les valeurs permettant de ramasser des cartes sont 2, 5, 6 plus la somme $2+5=7$, $2+6=8$, $5+6=11$ n'existant sur une seule carte et donc impossible. Il est donc nécessaire à ce joueur d'avoir 2, 5, 6, 7 ou dame. Comme il n'a aucune de ces cartes, il est obligé d'étaler simplement. Il «choisit» le mot est d'importance puisque l'on programme de poser le 4. Avec cette carte en plus, de nouvelles possibilités apparaissent : le 4 et la valet et le roi.



Mais maintenant c'est de nouveau au premier joueur de déposer une carte et avec son sept, il s'empare du 5 et 2, ce qui donne :



Le deuxième joueur reprend la main et avec son roi, récupère les deux dernières cartes laissant le tapis vide, il marque alors une «Schoba». Le premier joueur, ne pouvant rien ramasser ne peut que disposer son as suivi par son adversaire qui dépose à son tour son valet. Lorsque toutes les cartes ont été jouées, on redistribue à chacun trois nouvelles cartes, et l'on recommence jusqu'à épuisement du paquet. Le manche est terminé quand il ne reste plus de cartes à distribuer, à ce moment, le dernier joueur à avoir ramassé des cartes du tapis ajoute à son tas celles restant sur le tapis.



Decomptage des points

- Chaque joueur compte 1 point par Schoba survenue en cours de jeu. Puis quatre autres points peuvent être distribués à la fin de la partie : marque 1 point celui qui :
 - a le sept de carreau ;
 - a le plus de cameaux : il y en a 10 si chacun à 2 ou 41 en a 5. Il y a égalité et personne ne marque de point ;
 - a le plus de cartes : il y en a 40, a 20 égalité et pas de points ;
 - a le plus de sept, a égalité sur le sept le plus de six, a égalité pas de point.
- Ajouté aux «Schoba» ce total est le nombre de points marqué dans la manche. Une partie se joue en 21 points.

Stratégie : à vous de voir ce qu'il faut privilégier dans les cartes à ramasser. Vous vous doutez que le décomptage final est prééminent : un sept de carreau ne s'écroulera pas sur un tapis, il est inutile de courir après les cameaux ou les sept des que votre adversaire en a emporté respectivement 6 et 3. Il faut chercher à faire des combinaisons mais aussi à empêcher votre adversaire de faire les siennes et surtout faire des «Schoba» ! Aussi, fins stratégies à vos crayons et si la programmation n'aboutit pas, il vous restera la consolation de jouer entre amis ! Nous pensons que pour améliorer la stratégie de l'ordinateur, il vous faudra d'abord y jouer.

ERRATUM

Deux erreurs se sont glissées dans notre précédent numéro à la page 30.

- D'une part, l'équation :

$$ax^2 + bx^2 + cx + d = 0$$

peut se mettre sous la forme $X^2 + pX + q = 0$ avec $X = x - p/3$.

- d'autre part, le discriminant s'écrit :

$$\Delta = \frac{p^2}{27} + \frac{q^2}{4}$$

SEPTEMBRE 1986
Vient de paraître

BIBLIOTHÈQUE TECHNIQUE ÉDITIONS FRÉQUENCES

● INITIATION A LA VIDÉO LÉGÈRE (THÉORIE ET PRATIQUE)

Claude Gendre.

— Choix d'un standard ? — Caméscopes VHS VHS-C ou 8 mm ? — Connexion ? Compatibilité ? — Accessoires ? Montage ? Enfin... Comment filmer ?
Le nouveau livre de Claude Gendre répond à toutes ces questions. Cet ouvrage essentiellement pratique, qui n'a pas d'équivalent en librairie, aujourd'hui, a été révisé (sans formules mathématiques) et tous ceux passionnés (dès qu'ils ont vu) de vidéo ainsi qu'aux amateurs de belles images.

Des illustrations en couleur donnent une excellente idée des possibilités de « filmage » et de montage.
L'avenir du cinéma d'amateur et celui de la création par l'image passeront par la vidéo légère.
Ce livre devient urgent.

● LE 3^e TOME de INITIATION A LA MICRO-INFORMATIQUE (COURS 1^{er} CYCLE)

Claude Polgar. (Enfin paru !)

Non, on ne s'initie pas à la micro-informatique en 5 leçons !

Si vous croyez au Père Noël vous pouvez espérer apprendre l'informatique en lisant les innombrables « Cours de BASIC pour débutants » qui ont poussé comme des champignons dans les années 1980. Votre ordinateur risque de finir ses jours au-dessus de votre armoire.

Mais si vous voulez vraiment apprendre à programmer il faut avoir le courage de commencer par A pour arriver à Z. Programmer est un loup intelligent et peut devenir un métier passionnant, mais l'étude de la programmation nécessite un minimum de travail et de méthode.

Etre sérieux — c'est le pari que fit la revue LED-MICRO en publiant à partir de 1985 les 20 premiers cours de C. Polgar. Plus de 40 000 lecteurs les ont suivis. Ce succès nous a conduit à demander à C. Polgar de remettre son cours à jour et de le compléter. Le résultat est remarquable (2 tomes, plus de 700 pages format 21 x 27), permettent d'acquiescer rapidement des connaissances solides.

● INITIATION A L'ÉLECTRICITÉ ET A L'ÉLECTROTECHNIQUE

Roger Friedérich.

Vous trouverez aisément en librairie des ouvrages d'initiation à l'électronique ou aux techniques les plus avancées des circuits intégrés etc. Mais si vous désirez une initiation aux bases de l'électricité et de l'électrotechnique sans vous en remettre à des ouvrages scolaires, alors vous ne trouverez pas ! Nous avons demandé à un spécialiste de ces disciplines de tenter d'expliquer de la manière la plus claire tout ce qui se rapporte à l'électricité et ses applications ainsi qu'à l'électrotechnique. Il a réussi et nous sommes certains que dans ce domaine il fallait être reconnaissant par la loi d'Ohm et répondre à la question : Comment ça marche ?

Chaque mois, les nouveautés seront signalées.

**VOIR AU DOS NOTRE COLLECTION COMPLÈTE AINSI QUE LES PRIX DE CHAQUE
OUVRAGE ET SES CARACTÉRISTIQUES**

Diffusé après des librairies agréées exclusivement par les Éditions Eyrolles.
Bon de commande à retourner aux Éditions Fréquences 1 Boulevard Ney 75018 Paris.
Je desire recevoir tel(s) ouvrage(s) ci-dessous référencé(s) que je coche d'une croix :

<input type="checkbox"/> E 01	<input type="checkbox"/> E 02	<input type="checkbox"/> E 03	<input type="checkbox"/> E 04	<input type="checkbox"/> E 05	<input type="checkbox"/> E 06	<input type="checkbox"/> L 07	<input type="checkbox"/> P 08	<input type="checkbox"/> L 09	<input type="checkbox"/> L 10
<input type="checkbox"/> L 11	<input type="checkbox"/> E 12	<input type="checkbox"/> E 13	<input type="checkbox"/> L 14	<input type="checkbox"/> E 15	<input type="checkbox"/> P 16	<input type="checkbox"/> P 17	<input type="checkbox"/> P 18	<input type="checkbox"/> P 19	<input type="checkbox"/> L 20
<input type="checkbox"/> P 21	<input type="checkbox"/> E 22	<input type="checkbox"/> P 23	<input type="checkbox"/> P 24	<input type="checkbox"/> E 25	<input type="checkbox"/> P 26	<input type="checkbox"/> P 27	<input type="checkbox"/> P 28	<input type="checkbox"/> P 29	

Préciser par : ☐ 12 F par titre commandé, soit la somme totale ci-jointe de Frs ☐ par CCP ☐ Chèque bancaire ☐ Mandat-lettre ☐

Nom

Prénom

Adresse

Ville

Code postal



BIBLIOTHEQUE TECHNIQUE

Collection études (format 165 x 240)



LES SYNTHETISEURS
UNE NOUVELLE TECHNIQUE...
E 15, 135 p. Prix: 140 F TTC
Toute les applications spectrales des synthétiseurs, quelle qu'elle soit, sont traitées en fonction d'un langage commun. L'ouvrage est illustré de nombreux schémas et de nombreux exemples de programmes. Il est très utile pour les étudiants et les professionnels de l'électronique.



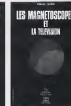
LES HAUT-PARLEURS
E 21, 120 p. Prix: 160 F TTC
Les haut-parleurs sont un élément essentiel de tout système audio. Ils sont donc très intéressants à étudier. Cet ouvrage traite de tous les aspects de leur conception, de leur utilisation, de leur entretien, de leur réparation.



INTRODUCTION A L'AUDIO-NUMERIQUE
E 25, 100 p. Prix: 120 F TTC
C'est le premier ouvrage dans ce domaine technique. Il est très utile pour les étudiants et les professionnels de l'électronique.



L'OPTIMISATION DES HAUT-PARLEURS ET ENCEINTES ACOUSTIQUES
E 24, 140 p. Prix: 120 F TTC
Cet ouvrage traite de l'optimisation des haut-parleurs et des enceintes acoustiques. Il est très utile pour les étudiants et les professionnels de l'électronique.



LES MAGNETOSCOPES ET LA TELEVISION
E 23, 150 p. Prix: 120 F TTC
Cet ouvrage traite de la télévision et des magnétoscopes. Il est très utile pour les étudiants et les professionnels de l'électronique.



PERIPHERIQUES: INTERFACES ET TECHNIQUE
E 22, 130 p. Prix: 120 F TTC
Cet ouvrage traite de la technique des périphériques et des interfaces. Il est très utile pour les étudiants et les professionnels de l'électronique.



L'ELECTRONIQUE DES MICRO-ORDINATEURS
E 26, 120 p. Prix: 120 F TTC
Cet ouvrage traite de l'électronique des micro-ordinateurs. Il est très utile pour les étudiants et les professionnels de l'électronique.



LES MAGNETOPHONES
E 20, 100 p. Prix: 80 F TTC
Cet ouvrage traite des magnétophones. Il est très utile pour les étudiants et les professionnels de l'électronique.



SELECTION DE L'AUDIOPHILE
E 12, 250 pages. Prix: 150 F TTC
Cet ouvrage traite de la sélection de l'audiophile. Il est très utile pour les étudiants et les professionnels de l'électronique.



LE MINI STUDIO
E 25, 100 pages. Prix: 140 F TTC
Cet ouvrage traite du mini studio. Il est très utile pour les étudiants et les professionnels de l'électronique.

Collection loisirs (format 135 x 210)



conseils et tours de main en électronique
E 27, 140 p. Prix: 40 F TTC
Cet ouvrage traite des conseils et des tours de main en électronique. Il est très utile pour les étudiants et les professionnels de l'électronique.



les lecteurs de compact-disco
E 28, 100 p. Prix: 120 F TTC
Cet ouvrage traite des lecteurs de compact-disco. Il est très utile pour les étudiants et les professionnels de l'électronique.



le langage de l'électronique anglo-français
E 29, 120 p. Prix: 80 F TTC
Cet ouvrage traite du langage de l'électronique anglo-français. Il est très utile pour les étudiants et les professionnels de l'électronique.



littres acides et passifs pour enceintes acoustiques
E 30, 120 p. Prix: 80 F TTC
Cet ouvrage traite des littres acides et passifs pour enceintes acoustiques. Il est très utile pour les étudiants et les professionnels de l'électronique.



17 montages électroniques
E 31, 120 p. Prix: 80 F TTC
Cet ouvrage traite de 17 montages électroniques. Il est très utile pour les étudiants et les professionnels de l'électronique.



WEEK-END PHOTO
E 32, 120 p. Prix: 120 F TTC
Cet ouvrage traite de la photographie. Il est très utile pour les étudiants et les professionnels de l'électronique.

DES EDITIONS FREQUENCES

Collection initiation (format 210 x 270)



P 08 95 pages. Prix: 115 F TTC
Ce ouvrage est un ouvrage d'initiation à la robotique. Il est destiné à tous les amateurs de robotique. Il est divisé en deux parties. La première partie est consacrée à la robotique et la seconde partie est consacrée à la programmation.



P 16 212 pages. Prix: 130 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 17 256 pages. Prix: 130 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 27 290 pages. Prix: 130 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 18 130 pages. Prix: 85 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 19 104 pages. Prix: 85 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 21 130 pages. Prix: 120 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 22 130 pages. Prix: 120 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 34 90 pages. Prix: 130 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 35 152 pages. Prix: 150 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 36 110 pages. Prix: 130 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.



P 37 110 pages. Prix: 130 F TTC
Ce livre est un ouvrage d'initiation à la micro-informatique. Il est destiné à tous les amateurs de micro-informatique. Il est divisé en deux parties. La première partie est consacrée à la micro-informatique et la seconde partie est consacrée à la programmation.

C'EST ARRIVE DEMAIN



(en direct de notre envoyé permanent dans la Silicon Valley)

Ce mois-ci, j'aurais pu appeler cette rubrique «C'est arrivé hier». En effet, il s'est produit un événement important récemment, et notre fierté nationale peut être flattée. La ville de Boston a créé voilà quelques mois un musée de l'ordinateur. Jusqu'à présent, seuls des gros systèmes y étaient représentés. Mais, à l'initiative d'une revue de micro, des systèmes personnels y ont été ajoutés. Parmi eux, une mini compétition a été organisée, pour juger quels étaient les meilleurs, respectivement. Parmi les quatre gagnants, sur 200 micros représentés, le légendaire MICRAL, micro-ordinateur français, le premier micro en fait. Pour que des américains, avec parmi eux Steve Wozniak, père de l'Apple II, choisissent un micro qui ne soit pas d'Outre-Atlantique, il faut que cette machine soit vraiment une œuvre digne de l'histoire des sciences et techniques. Quel dommage que le seul micro français vendu de nos jours per Bull soit en fait à 100 % américain.

Une des grandes idées de ces dernières années est de mettre à l'actif de M. Teller, illustre inconnu peut-être, mais génial. Nous lui devons les premières boutiques de location de logiciels. Cela signifie que pour une modeste partie du prix réel du logiciel, il est possible de tester un programme, de vérifier s'il correspond bien à ses besoins ou souhaits, puis si l'on est décidé, de l'acheter au prix habituel, moins la location. Je vous le disais, génial. Hélas, trois fois hélas, les gros fabricants de logiciels, et les gros distributeurs ont pris ombrage de cette pratique. Ils ont donc persuadé, une femme sonneur de partir en croisade contre la location. Cette brave dame, complètement inconnue des milieux informatiques, et pour cause, est donc en guerre. Un projet de loi interdisant la location est donc sur le bureau du Sénat américain, et

sera sans doute voté. Pourquoi. Officiellement parce que la location favorise le piratage, et diffuse pour une durée limitée auprès du plus grand nombre des programmes peu ou pas protégés. En effet, conjointement, la protection est en voie de disparition ici. Ceci implique donc que les fabricants et distributeurs considèrent leurs clients potentiels comme des voleurs. Mais alors, pourquoi retirer la protection ? Parce que les clients disposant de disques durs, qui se répandent, refusent simplement d'acheter des programmes qui ne pourront être installés sur ce disque. Entre deux maux, choisissant le moindre, les gros bonnets du logiciel retirent les protections, mais veulent tout de même la location. De plus, ils mettent en parallèle la location de disquettes et celle de disques, qui est interdite ici. Comme si l'on pouvait copier un programme et une chanson. Mais, et éternel l'aveugement jamais, le vrai problème n'est pas là. En fait, ils craignent que le simple essai de la plupart des gros programmes ne suffise aux acheteurs de se rendre compte de leurs faiblesses. En effet, de plus en plus, les publicités sont carrément mensongères, et dans ce cas, il est ample de s'en rendre compte. Comment fait-on pour mentir ? Les méthodes sont simples :

- telle fonction décrite sur la publicité n'existe pas en fait. C'est parce qu'elle doit apparaître sur la prochaine version, attendue sous peu. En fait, la version suivante se fait attendre, n'apparaît jamais, ou encore ne dispose pas de cette fonction.
- des temps d'exécution sont faux, ou pour le moins mensongers
- vous voyez sur la publicité un écran qui ne correspond pas en fait à votre ordinateur, mais au Mac, ou au plus gros des PC.

Il est évident que quelques minutes d'essai réel sont alors un révélateur sans prix. Hors, ces pratiques

abusives sont toujours le cas des grosses sociétés en question. Celle-ci ont donc tout intérêt à empêcher la diffusion de ces lacunes par des voies autre que profitables que la location. En outre, les sociétés proposent des bons programmes, qui sont souvent créées pour un programme, se trouvent tout à fait satisfaites de la diffusion de ceux-ci par cette voie. Le partage ne semble pas les effrayer, et la publicité a bon marche due à la location leur paraît bonne à prendre.

Ainsi, faut-il pondre une loi qui protège les mauvais programmes, en défavorisant les bons, des petites sociétés. Mme le sénateur en semble persuadée. Mais les journaux d'informaticiens de tout le pays sont partis en guerre contre ce projet de loi, le lobby des fabricants est très puissant.

Une petite société vient de proposer un programme assez intéressant pour les gens qui en ont assez de devoir taper les lettres ou des rapports dans un langage stéréotypé. C'est un traitement de textes qui permet d'utiliser l'alphabet de 41 langues européennes, asiatiques, russes ou hébraïques. Il y a tous les caractères accentués, la mise en page, l'écriture de droite à gauche, de bas en haut...

C'est le premier programme de ce type et, il faut le remarquer, il est très bien fait. La documentation est écrite en plusieurs langues, le programme est simple et complet. Une belle réussite de programmation, qui devrait en être une commerciale.

Un autre domaine où la lutte entre revues et fabricants fait rage est le service après-vente. En effet, rien n'est plus aléatoire que ce service-ci. Vous pouvez acheter certains programmes très coûteux (500 \$ et plus) sans être assuré d'un SAV, ou alors le payer un prix prohibitif. De même, pour des programmes à 20 \$, vous avez un SAV au point. Tout d'accord, comment fonctionne un bon SAV ? Il faut avoir les versions nouvellement sorties gratuitement, il faut un service de réponses par téléphone gratuit, et efficace. Enfin, il faut former les vendeurs des boutiques vendant le produit. Reprenez ces points en détail. La nouvelle version gratuite : il est fréquent que certains programmes évoluent. Certains par exemple, nous l'indiquons, ne sont pas conformes à la publicité qui en avait été faite, d'autres comportent des erreurs de programmation, des bugs. Il n'est pas logique de payer pour les corrections, alors que vous avez déjà payé pour un programme qui plante sur certaines fonctions non déboguées. Hier certains fabricants demandant 10 \$, ou plus pour ce service, qui semble pourtant la moindre des choses. Imaginez l'achat d'un livre dont des pages seraient mal imprimées, et que non seulement on ne vous le reprenne pas, mais que lorsque la faute est comble, on vous refuse de payer. Le service par téléphone : la plupart des sociétés de logiciels le propose, même pour des petits programmes. Ce service permet de disposer d'une aide à votre programme plante sans que vous ne compreniez où et

pourquoi, ou pour vous aider à réaliser des applications professionnelles. Cela constitue en fait un complément indispensable à une documentation souvent non exhaustive, et bâclée. Certaines sociétés proposent ce service gratuit, par un numéro vert, mais d'autres le facturent, et parfois à des tarifs carrément déraisonnables, de plusieurs centaines de dollars l'heure. Qui plus est, il est hélas fréquent que le technicien que vous parviendrez après bien des problèmes à joindre, soit totalement incompétent. Il ne sait pas, croit que, pense que peut-être... C'est désespérant, et improductif. Le service en boutique quant à lui est le plus souvent totalement incapable de vous dire autre chose que ce qui est sur la boîte, n'ayant jamais imaginé de l'aider pour apprendre quoi que ce soit sur le programme. Alors, si vous en savez plus que le vendeur, n'espérez rien de lui. Cette situation intolérable est la plus fréquente.

Pour lutter contre toutes ces lacunes, les revues US se sont livrées à des essais comparatifs sans tentatives de la plupart des distributeurs de logiciels, et de leurs programmes. Les résultats sont souvent attendus, mais certains se font malgré tout fort bien. En effet, des sociétés proposent un soutien téléphonique gratuit, des releases gratuites automatiques et une mise à jour des vendeurs (ce dernier point est tout de même le plus rare). D'autres au contraire ne font absolument rien. Ainsi certaines sociétés font payer le prix de la consultation téléphonique, pour obtenir finalement un technicien ignorant tout du programme en question, non formé à ces problèmes, et que cela n'intéresse pas. Au bout du compte, le SAV est inutile et coûteux, car les communications trans américaines sont fort onéreuses. Aussi, un critère d'achat est-il devenu le SAV et le soutien téléphonique gratuit.

Il est de plus en plus évident que l'ordinateur est devenu une magnifique petite machine à produire des sons. Vous connaissez sans doute la norme MIDI, qui officialise l'entrée de la micro dans le monde de la musique. Depuis sa sortie, il n'est de semaines sans de nouveaux accessoires pour permettre aux musiciens, parfois expérimentés, de développer leur art. Il est possible de relier des guitares, des orgues, des synthétiseurs. Des professionnels se mettent à ne produire que des morceaux créés par un générateur logiciel. Le plus évolué actuellement est le Macintosh, dont le petit synthétiseur intégré est une bonne base. Mais, la plupart des ordinateurs familiaux sont maintenant équipés de synthétiseurs, ou d'interface MIDI. Le pionnier avait été Yamaha, bien connu pour ses instruments de musique, ceci expliquant cela.

Il ne faut pas croire que cela simplifie la création de morceaux. J'ai sous les yeux au moment où je tape cet article des musiciens en herbe dont on se dit qu'ils ne feront rien de plus qu'avec un piano, et c'est fort peu.

COURS D'INITIATION AU PROGICIEL MULTIPLAN

Charles-Henry Delaleu

2^e PARTIE

Voici le deuxième cours sur le progiciel Multiplan qui termine les bases de ce tableau. Le mois prochain, nous attaquerons les commandes évoluées. Comme pour tout apprentissage d'un logiciel, il est souhaitable d'agrémenter la théorie avec un maximum de pratique. Il est donc conseillé aux lecteurs disposant d'une machine et d'un Multiplan d'essayer au fur et à mesure du cours toutes les commandes et fonctions décrites. Nous agrémenterons le prochain cours d'un maximum d'exemples. En première partie, nous avons abordé :

- La présentation sommaire du Multiplan
- L'écran de commande
- Le déplacement du pointeur de cellule
- La ligne état
- Le défilement de la feuille de calcul
- Les commandes clavier

Aujourd'hui, nous étudierons :

- La commande VERS
- La commande GUIDE OPERATEUR
- La commande TAB
- La commande QUIT
- La commande ALPHA
- La commande LARGEUR COLONNE
- La commande CALCUL
- La commande FORMAT
- Les sauvegardes
- La commande INSER
- Le caractère
- La commande BLANC
- La commande RECOPIE
- L'EXPRESSION - CALCUL

La commande VERS

Nous avons observé qu'il était possible de se déplacer d'une cellule à l'autre en utilisant les flèches de direction. La commande VERS autorise un déplacement direct à la cellule désirée grâce à la fonction «Ligne-Col». Il suffit alors de préciser le numéro de ligne et le numéro de colonne souhaités.

La commande GUIDE OPERATEUR

VERS

Sert à déplacer le pointeur de cellule sur la feuille

VERS LIGNES COLONNES

Déplace le pointeur de cellule directement à la ligne et à la colonne spécifiées. Si la cellule demandée est déjà visible sur l'écran, celui-ci ne sera pas déplacé. Dans le cas contraire, l'écran est déplacé pour visualiser la cellule spécifiée.

VERS MOI

Déplace le pointeur de cellule directement au sein supérieur gauche de la zone désignée. Les touches de direction peuvent servir pour visualiser indépendamment le répertoire de zone.

VERS ZONE RECTANGULAIRE

Déplace la feuille afin que la cellule désignée soit en coin supérieur gauche de la fenêtre désignée.

GUIDE OPERATEUR: Répertoire Début, Suivant, Précédent,
Application, Commandes, Edition, Ajout, Suppression, Touches
Chaque fois que vous appuyez sur la touche de commande
L156C3 100% Libre Multiplex TEMP

La commande GUIDE est utilisée comme une assistance. Si une fonction vous paraît confuse, la commande GUIDE vous rafraîchit la mémoire.

La commande TAB

Grâce à la touche de tabulation (TAB) du clavier de votre micro-ordinateur, vous pourrez vous déplacer dans le sens même de chacune des fonctions d'une commande. Ainsi, dans le cas de la fonction GUIDE, vous pourrez vous déplacer sur les routines: Répertoire (retour en mode normal), Début (début du guide), Suivant (page suivante), Précédent (page précédente), etc. Chaque fois que l'on appuie sur la touche TAB, on passe à la routine suivante.

La commande QUITTE

QUITTER

Confirme par O (oui)
L156C3

100% Libre Multiplex TEMP

La commande QUITÉ termine la session de travail sous Multiplan. Elle doit être confirmée par un OUI. En effet, cette commande n'effectue pas une sauvegarde des tâches réalisées. Pour toute sauvegarde, utiliser la commande LIT-ÉCRIT.

La commande ALPHA



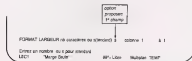
Bien que Multiplan soit un tableur spécialisé dans les nombres et les calculs, il est souhaitable d'agencer sa feuille de calcul d'un certain nombre de mots et d'expressions alphabétiques afin d'améliorer la lisibilité du travail effectué. Il ne faut donc pas hésiter à employer des titres, des explications, etc. Pour être en mode Alpha, il suffit, sous le menu principal, de pointer la commande Alpha à l'aide de la touche TAB ou d'appuyer sur la touche A du clavier.

La position des mots à placer sur la feuille est déterminée par la position de la cellule active. Pour les corrections, il suffit d'utiliser les touches Retour Arrière et la barre d'espace.

La commande LARGEUR DE COLONNE



Si la taille d'une cellule est trop étroite pour placer la chaîne de caractères que vous desirez, il suffit d'utiliser la commande FORMAT (commande Largeur).



Une colonne peut être remplacée en mode standard

FORMAT LARGEUR (à évaluer ou standard) 15 colonne 1 à 1

Entrez un nombre ou 0 pour standard

LOC1 Marge Gauche 80% Large Multiplicateur TEMP

La programmation de la largeur d'une cellule se fait par colonnes entières. Il convient d'indiquer le nombre de caractères souhaités, puis son numéro et à défaut une extension.

Ex Pour formater la colonne 1 INDICHER Colonne 1 à 1

Pour formater les colonnes, de la colonne n° 7 à la colonne n° 12 INDICHER Colonne 7 à 12

La commande CALCUL

a. Les nombres

Appuyez sur la commande
indique "calcul"
Après les chiffres 0 à 9

CALCUL 20000

Entrez une expression

LOC2 80% Large Multiplicateur TEMP

Pour entrer un nombre, le processus est similaire à l'alphanumérique. Ici, on frappe C au lieu de A. Une fois le nombre placé, il suffit d'appuyer sur ENTER.

Attention: pour un Multiplan version française, la virgule est conforme au système français de représentation des nombres.

BON	FALIX
1 275,52	1 275.52

b. Les calculs

	1	2	3
1	ECRAN	1 500,00	
2	UNITE CENTRALE	12 475,22	
3	CLAVIER	856,27	
4	TOTAL	?	
5			

Les opérations simples se réalisent de la manière suivante sur Multiplan : soit l'addition à effectuer (prix du clavier + prix de l'unité centrale + prix de l'écran)

Position des cellules

Prix de l'écran	1 500,00	L1 C2
Prix de l'unité centrale	12 475,22	L2 C2
Prix du clavier	856,27	L3 C2
TOTAL	?	L4 C2

l'addition donnant le total s'écrit : L1C2 + L2C2 + L3C2

- Marche à suivre :
1. Placer la cellule active en L4C2
 2. Activer la commande CALCUL
 3. Ecrire l'addition
 5. Taper sur ENTER (validation de l'opération)

La commande FORMAT

FORMAT Cellules Standard Options Largeur

Choisissez une option ou appuyez le caractère de commande

L4C2 15000 50% Libre Multiplan TEMP

Il est possible de formater la présentation des cellules suivant le tableau situé en bas de page : taper la commande FORMAT

FORMAT STANDARD Cellules Largeur

Choisissez une option ou appuyez le caractère de commande

L4C2 15000 50% Libre Multiplan TEMP

Choisissez l'option «Cellules»

FORMAT STANDARD CELLULES alignement: Cr Norm Gauche Droite

code format: Caractère Puissance/Signe P + %

Nb de décimales: 0

Choisissez une option

L4C2 15000

50% Libre

Multiplan TEMP

Grâce à la fonction TAB, définissez la présentation désirée

Nota : Cette commande est double, elle autorise

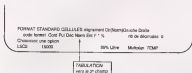
- le mode d'alignement
- le code Format

Code Format

Définition	Signification	Exemples
Cont	Continuation	Établissements Martin
Pu	Notation scientifique	1,4301E-23 4,87E5
Déc	Nombre décimal	4 613
Norm	Normal	le texte et les nombres sont présentés en format normalisé
Ent	Numéro entier	3,1416 représente par 3
F	Francs	20 000,00 F (150,00 F)
*	Graphique à barres	3 représente par ***
%	Pourcentage	0.0613 représente par 6.13 %
-	(ne pas changer le format)	

Le code Format autorise une présentation plus nette des informations contenues à l'écran. De plus, les pourcentages peuvent être calculés rapidement

La fonction alignement



La fonction alignement permet d'obtenir une meilleure cohérence des colonnes de la feuille de calcul

Définition	Exemples	Résultat
Cont		le texte les chiffres sont centrés
Norm	Ventes 1 000,25 50,25	le texte est justifié à gauche, les nombres à droite
Gauche	Ventes 1 000,25 50,25	le texte et les nombres sont justifiés à gauche
Droite	Ventes 1 000,25 50,25	le texte et les nombres sont justifiés à droite

Les sauvegardes

LIT_ECRIT Charge Sauvegarde Efface Detruit Option Remome

Choisissez une option ou appuyez le caractère de commande

LSC2 10000 88% Libre Multplan TEMP

Nous avons vu que la commande QUITTE permet de sortir d'une session Multplan. Toutefois elle peut être dangereuse car il n'y a pas de sauvegarde du travail réalisé. La commande LIT-ECRIT donne l'accès à l'archivage.

- CHARGE** : Cette routine charge en mémoire centrale une feuille de calcul stockée en mémoire de masse (disque).
- SAUVEGARDE** : Cette routine sauvegarde en mémoire de masse le contenu de la feuille de calcul.
- EFFACE ECRAN** : Efface après confirmation entièrement la feuille de calcul.
- DETRUIT** : Detruit après confirmation une feuille contenue en mémoire de masse.
- OPTION** : Indique le repertoire ou le disque à utiliser pour les chargements et les sauvegardes.
- RENAME** : Redonne un nom à une feuille contenue en mémoire de masse.

Exemple : la sauvegarde

- 1 Choisir la commande LIT-ECRIT
- 2 Pointer la routine Sauvegarde (à l'aide de TAB)
- 3 Indiquer le nom que vous voulez donner à votre feuille (FICHIER)
- 4 Taper sur ENTER

La commande INSERE

INSERE Ligne Colonnes

Choisissez une option ou appuyez le caractère de commande

LSC2 88% Libre Multplan MARTIN

Il peut arriver, pour diverses raisons, de vouloir insérer des informations entre des données déjà existantes. La commande INSERE permet de créer de la place entre des lignes et des colonnes déjà existantes. Le processus est le suivant :

INSERE LIGNE n° de ligne 7
entre colonne 1

Entrez un nombre

LSC2

avant ligne 8
et 82

88% Libre

Multplan MARTIN

Exemple Insertion de 7 lignes

En ligne 8 se trouve le terme «marge brute», en ligne 5 le terme «Cont». Nous voulons placer 7 nouvelles lignes entre ces deux mots. Nous choisissons la routine INSERE LIGNE.

- a. Il faut indiquer le nombre de lignes desiré → 7
- b. La position où l'on désire placer ces lignes → avant la ligne 6
- c. Il est nécessaire de préciser les références des colonnes concernées → colonne 1 à colonne B3.

#1	1	2	3	4	5	6
1						
2						
3	Montre	20000 08 F				
4						
5	Cash	15000 08 F				
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16	Marge Brute					
17						
18						
19						

7 nouvelles lignes ajoutées

COMMAND: Alpha Beta Calc Delvrt Edit Format Guide Inters Lt, Rst Move
 Next Options Prtgte Quidt Respur Scler Tr Vals Xlats ZonPrtmtr
 Choisissez une option ou appuyez le caractère de commande.
 LACC 19F Ligne Multplan MARCH

Une fois les choix réalisés, il suffit d'appuyer sur ENTER pour que le contenu de la ligne 8 passe en ligne 15. Ainsi, un espace est créé afin d'insérer de nouvelles lignes. Le processus est similaire pour insérer des colonnes.

Nota : Dans le cas de l'utilisation de la commande INSERE, vérifier que l'ensemble des présentations des cellules soit identique (commande FORMAT).

Le caractère « : »

Dans Multplan, le caractère « : » est une sorte de joker. Il permet d'exécuter des opérations sur plusieurs cellules à la fois.

Exemple : Dans le cas de la commande FORMAT, je désire formater une cellule. Il suffit alors de la pointer et de demander la présentation requise. Si je veux étendre cet ordre à plusieurs cellules en même temps, j'utilise alors le caractère « : ».

A l'instant où je demande la commande FORMAT, je me trouve en L17C22. Je désire présenter un ensemble de nombres placés entre les lignes 17 et 30 et compris entre les colonnes 22 et 50. Grâce au mode TAB, je me place sur l'adresse de la cellule pointée. Je tape alors à la place L17C22 la référence suivante :

L 17 : 30 C 22 : 50

lignes 17 à 30

colonne 22 à 50

Ainsi l'ensemble des cellules comprises dans mon ordre sera formaté suivant le mode choisi.

La commande BLANC

BLANC remise à blanc cellules LSC2

Entrez une référence de cellule ou un groupe de cellules

LSC2 10002 99% Libre Multiplan MARTIN

Une cellule déjà programmée peut être remise à blanc (remise à blanc = effacer son contenu). La commande BLANC permet cette opération. Bien entendu le mode « » peut être utilisé pour effacer un certain nombre de cellules.

Ex. BLANC remise à blanc cellules L 5 C 2 (1)
 BLANC remise à blanc cellules L 12 24 C 3 (2)
 BLANC remise à blanc cellules L 8 9 C 22 35 (3)
 BLANC remise à blanc cellules L 5 C 2 - 5 (4)

1. Effacer la cellule LSC2
2. Effacer les cellules de la colonne 3 comprises de la ligne 12 à la ligne 24
3. Effacer les cellules de la colonne 22 à 35 et de la ligne 8 à 9
4. A vous de trouver !

La commande RECOPIE

RECOPIE Déplace Vers la Droite Cellules

Choisissez une option ou appuyez le caractère de commande

LSC2 20000 99% Libre Multiplan MARTIN

Multiplan autorise la recopie du contenu d'une ou plusieurs cellules (nombres, chaînes de caractères, équations)

RECOPIE DROITE de de cellules 11 depuis LSC2

Entrez un nombre

LSC2 20000 99% Libre Multiplan MARTIN

Exemple d'une recopie vers la droite de 11 cellules depuis l'adresse LSC2

MAX(Liste)	Donne la valeur du plus grand élément de Liste
MIN(Liste)	Donne la valeur du plus petit élément de Liste
MOYENNE(Liste)	Calcule la moyenne des valeurs de Liste. Elle donne le même résultat que l'expression $SOMME(Liste)/NB(Liste)$

Fonctions mathématiques, logiques et textes

ABS(N)	Donne la valeur absolue de l'argument N
ARRONDIR(DEC)	Donne une valeur arrondie de N avec un nombre de décimales spécifié par DEC
ATAN(N)	Calcule la fonction arctangente de l'argument donnant un angle en radians
CHERCHE(N,Table)	Si TABLE (un groupe rectangulaire de cellules) est plus haut que large, MULTIPLAN va chercher, dans la première colonne, la ligne contenant une valeur inférieure ou égale à N. Le résultat de la fonction sera la valeur contenue dans la dernière cellule de la ligne ainsi définie. Si TABLE est plus large que haut, les rôles des lignes et des colonnes sont inversés.
COLONNE()	Donne le numéro de la colonne dans laquelle se trouve la formule contenant cette fonction
COS(N)	Calcule le cosinus de l'argument exprimé en radians
CTXT(N,Decimales)	Convertit la valeur spécifiée en un texte représentant un nombre décimal comprenant le nombre de décimales indiquées par le second argument. L'action de cette fonction est équivalente à celle du code de format "Dec."
ENT(N)	Donne la partie entière du paramètre
ERREUR(Valeur)	Donne la valeur logique VRAI si l'argument se trouve être l'une quelconque des valeurs d'erreurs
EXP(N)	Calcule la valeur de la fonction exponentielle correspondant à la valeur du paramètre N
FAUX()	Donne la valeur logique FAUX
FRANCON(decimale)	Convertit l'argument N en un texte représentant la valeur en francs, suivi des caractères " F."
INDEX(Zone,Indices)	Donne la valeur d'une cellule sélectionnée par les indices d'une zone rectangulaire
LIGNE()	Donne le numéro de la ligne où l'expression contenant cette fonction apparaît
LN(Valeur)	Donne la valeur logique VRAI si l'argument est N/A! Dans le cas contraire, elle donne FAUX
LOG(N)	Calcule le logarithme naturel de l'argument
LOG10(N)	Calcule le logarithme base 10 de l'argument
MOD(Dividende,Diviseur)	Donne le reste de la division entière Dividende divisé par le Diviseur. Dividende et Diviseur doivent être positifs
NA()	Donne la valeur spéciale N/A (non accessible)
NBCARCT)	Donne le nombre de caractères contenus dans le texte
NON(Logique)	Donne un résultat qui est l'opposé de la valeur logique de l'argument
NUM(T)	Le texte T doit contenir la représentation d'une valeur numérique. La valeur de cette constante est le résultat de la fonction

RACINE(N)	Donne la valeur de pi (3,14159...)
REPT(T,N)	Donne la racine carrée de l'argument
SI(Logique,Alors Valeur1,Sinon Valeur2)	Donne un texte se composant de N répétitions du texte T.
	Si Logique est VRAIE la fonction donne Valeur1, si elle est FAUX, elle donne Valeur2
SIGNE(N)	Donne un nombre représentant le signe algébrique de l'argument
SIN(N)	Calcule le sinus de l'argument exprimé en radians
STXT(T,Depart,Nombre)	Donne le sous-texte extrait à partir du texte T. Depart indique la position dans T du premier caractère du sous-texte et Nombre indique le nombre de caractères du sous-texte
TAN(N)	Calcule la tangente de l'argument exprimé en radians
VRAI()	Donne la valeur logique VRAI

Valeurs d'erreurs

DN/DI	Division par 0.
N/A?	Donnée non accessible
NOM?	Nom non défini
NUM?	Dépassement de capacité ou erreur numérique
REF?	Référence à une cellule non existante
RIEN?	Intersection de zones disjointes
VALEUR?	Utilisation de valeur de type incorrect
!!!!	Nombre trop long pour tenir dans la colonne

ABONNEZ-VOUS A



Je désire m'abonner à **LED-MICRO** France : 160 F - Etranger* : 240 F

NOM

PRENOM

N° RUE

CODE POSTAL VILLE

* Pour les expéditions « par avion » à l'étranger, ajoutez 60 F au montant de votre abonnement.

Joins mon règlement par : chèque bancaire ☐ C.C.P. ☐ Mandat ☐

Le premier numéro que je désire recevoir est : N°

Le cours d'initiation le plus complet + de 700 pages



Non, on ne s'initie pas à la micro-informatique en 5 leçons !

Si vous croyez au Père Noël vous pouvez espérer apprendre l'informatique en lisant les innombrables «Cours de BASIC pour débutants» qui ont poussé comme des champignons dans les années 1980. Votre ordinateur risque de finir ses jours au-dessus de votre amorce.

Mais si vous voulez vraiment apprendre à programmer il faut avoir le courage de commencer par A pour arriver à Z. Programmer est un loisir intelligent et peut devenir un métier passionnant, mais l'étude de la programmation nécessite un minimum de travail et de méthode.

Etre sérieux - c'est le pari que fit la revue LED-MICRO en publiant à partir de 1985 les 20 premiers cours de C. Polgar. Plus de 40 000 lecteurs les ont suivis. Ce succès nous a conduit à demander à C. Polgar de remettre son cours à jour et de le compléter. Le résultat : un ouvrage épais (3 tomes, plus de 700 pages format 21 x 27), permettant d'acquiescer agréablement des connaissances solides.

Diffusion auprès des librairies assurée exclusivement par
les Editions Eyrolles

Initiation à la micro-informatique C. Polgar

Bon de commande à retourner aux Editions Frequences
1, boulevard Ney 75018 Paris.

Je désire recevoir le tome 1 ☐ 140 F (130 F + 10 F de frais de port)
le tome 2 ☐ 140 F (130 F + 10 F de frais de port)
le tome 3 ☐ 200 F (180 F + 10 F de frais de port)

Ci-joint mon règlement par

☐ CCP ☐ Cheque bancaire ☐ Mandat

Nom

Prénom

Adresse

Code postal

Ville

Une seule
parmi près de 600 lettres
de lecteurs :



Initiation à la Micro-Informatique 1^{er} Cycle Tome 3

(enfin paru !)

3.16 (Suite et fin) L'affichage

- Étude des instructions permettant d'afficher des présentations « évoluées » : PRINT TAB - PRINT USING - LOCATE - COLOR en mode texte
- Présentation en tableaux de toutes sortes grâce à la pratique des opérateurs MODULO et DIVISION ENTIERE
- Beaucoup de programmes utilisent des assemblages de ces instructions et opérateurs - dont la combinaison n'est pas toujours facile

3.17 Compléments

- Étude des dernières instructions, fonctions et variables du cycle 1 : FILE#, KILL, AUTO, ON ERROR GOTO, RESUME, ERR, ERL, DELETE, EDIT, RENUM, TRON, TROFF, STOP, CONT, KEY ON, KEY OFF, FID, BEEP
- Compléments du cycle 1 qui sont maintenant accessibles aux élèves : sur la précision et les erreurs dues à l'arrondi, sur la sélection, les boucles

3.18 Graphisme

- Une étude complète et détaillée sur les instructions graphiques en haute résolution : SCREEN, PSET, PRESET, STEP, LINE, CIRCLE, COLOR, POINT, PAINT, sans éluder aucune des difficultés et « pièges » classiques (l'incrustation de texte dans le dessin, les « bavures » dues au PAINT mal utilisé...)
- Une étude détaillée du langage graphique DRAW avec ses subtilités et ses pièges (sous-chaînes X, paramètres variables dans le DRAW, etc.)
- De nombreux exercices avec leurs solutions BDI et leurs illustrations sur des photos d'œuvres en couleur (48 photos)

3.19. Dessin des courbes

- Un chapitre séparé du graphisme général (chapitre 3.18) de façon à ce que les « non mathématiciens » puissent le consulter sans remords - ils ne seront pas punis !
- Pour les mathématiciens, une excellente révision et illustration des courbes de toutes sortes : $Y = f(X)$, courbes paramétrées, courbes en coordonnées polaires avec des exemples utiles : courbes d'amortissement, astéroïde, cardiode, décomposition d'une fonction périodique par une série de Fourier

3.20. Révision générale

- L'enrichissement des notions selon l'ordre « pédagogique » qui a été utilisé jusqu'ici est bien différent de l'ordre « logique ». Autant qu'un cours d'anglais suit un ordre différent de celui qu'on trouve dans un grammaire anglaise
- Tout ce qui a été enseigné jusqu'ici résumé en 30 pages. Une référence pour retrouver la notion dont on a besoin à l'avance les cours et ses exercices. Mais aussi une référence sur la structure d'un langage informatique, d'où une prégnante note à la lecture des cours de PASCAL (par exemple !)

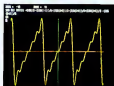
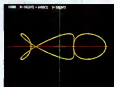
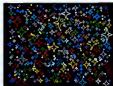
3.21. Techniques de mise au point

- Les outils de base : étude des logiciels de texte, de dessin et d'interprétation des messages d'erreur
- Comment organiser et corriger ses erreurs
- La représentation du dialogue homme-machine, pour noter l'expérience que vous acquièrez par la pratique

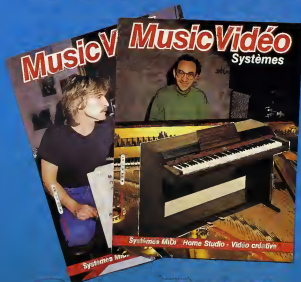
3.22. Problèmes de synthèse - Notions d'analyse

C'est à la fois la conclusion, le point le plus agréable et le plus utile du 66 cours. L'auteur ne se contente pas de fournir une liste de problèmes avec leur solution : il se met à la place du programmeur débutant en essayant de découvrir le « processus de réflexion » qui lui permet de l'énoncé d'un problème à sa solution (une échelle précoce à l'analyse)

1 livre broché de 248 pages pages 21 x 27, dont 8 pages en couleur



nouveau!



- exploiter toutes les possibilités des systèmes MIDI
- réaliser vous-mêmes un clip vidéo
- tirer le maximum de vos synthétiseurs
- installer chez vous votre studio d'enregistrement
- tout savoir sur les nouveautés musique et vidéo créatives

**Tout cela chaque mois
dans Music Vidéo Systèmes**

une publication des Editions Fragrances chez le livre marchand le 3^e trimestre
Editeurs Fragrances: 1, boulevard Ney 75 018 Paris - Tél. 45 77 41 77